PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-163266
(43)Date of publication of application : 16.06.2000

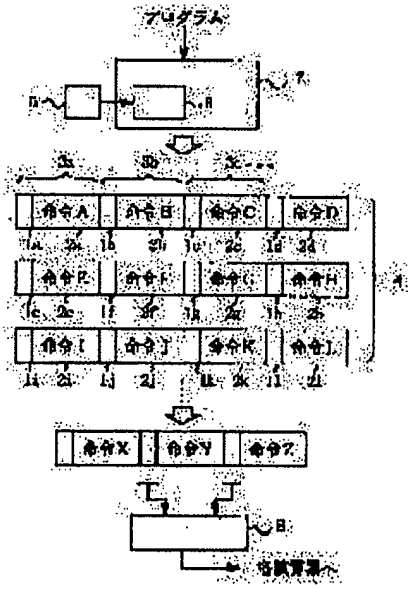
(51)Int.Cl. G06F 9/38
G06F 9/45

(21)Application number : 10-338834
(22)Date of filing : 30.11.1998

(71)Applicant : MITSUBISHI ELECTRIC CORP
(72)Inventor : HATTORI TAKASHI

(54) INSTRUCTION EXECUTION SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To enhance the use efficiency of a memory and to make executable instructions while maintaining the compatibility of the instruction sequence by judging whether or not each instruction needs to wait for the end of an instruction to be executed before based on decided register conflict decision information and executing the instructions in sequence or in parallel according to the judgement information in the instruction when the instruction are executed.



SOLUTION: A maximum parallel executable instruction number that a parallelism indicating means 5 is passed to a conflict decision means 6 that a compiler 7 has and the conflict decision means 6 judges the conflict relation of registers among four instructions which are possibly executed successively from the instruction number and reflects the jugement result on completion wait fields of respective instruction to generate an instruction sequence 4. An instruction executing means 8 fetches instructions A, B, and C as instructions X, Y, and Z and executes the instructions A to C in parallel since the completion wait fields of the instructions B and C at the positions of the instructions Y and Z all show that 'It is not necessary to wait for the completion of an instruction to be executed befor.'.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2000－163266
（P2000－163266A）
(43)公開日　平成12年６月16日(2000.6.16)

(51)Int.Cl.'		識別記号	FI			テーマコード'(参考)
G06F	9/38	370	G06F	9/38	370X	5B013
	9/45			9/44	322F	5B081

審査請求　未請求　請求項の数２　ＯＬ　（全６頁）

(21)出願番号　特願平10－338834

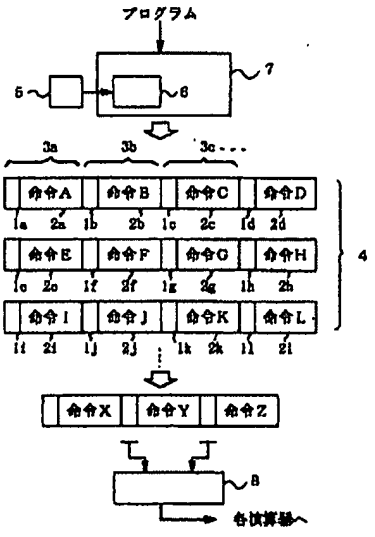
(22)出願日　平成10年11月30日(1998.11.30)

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号
(72)発明者　服部　孝
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
(74)代理人　100102439
弁理士　宮田　金雄　（外２名）

Ｆターム(参考)　5B013 DD01 DD04 DD05 DD10
5B081 CC32

(54)【発明の名称】　命令実行方式

(57)【要約】
【課題】　並列度（並列実行可能な命令数の上限）の異なる複数の計算機（命令処理装置）において、複数の命令を同時に並列実行する際に命令を再コンパイルすることなく、H／Wの増加をすることなく並列実行でき、命令列の互換性を保った命令実行ができる命令実行方式を提供する。
【解決手段】　命令の並列実行が可能な複数の計算機の中で最大の並列実行可能命令数と同数の命令単位で、各命令と各命令に先行する命令とが使用するレジスタの競合状態を判定し、この判定された情報に基づいて各命令が前に実行される命令の完了を待つ必要があるか否かを判断し、この判断情報を各命令中に設定し、命令実行時にこの各命令中の判断情報に基づいて各命令を実行するので、並列度の異なる複数の間で命令列の互換性を保った命令実行ができる。



1a, 1b, 1c,… :完了待ちフィールド
2a, 2b, 2c,… :命令フィールド
3a, 3b, 3c,… :命令
4：命令列
5：並列度指示手段
6：競合判定手段
7：コンパイラ
8：命令実行手段

1

【特許請求の範囲】

【請求項１】　命令の並列実行が可能な複数の計算機の中で最大の並列実行可能命令数を入力する並列度指示手段と、

前記並列度指示手段により入力された命令数と同数の命令単位で、各命令と各命令に先行する命令とが使用するレジスタの競合状態を判定する競合判定手段と、

前記競合判定手段で判定されたレジスタ競合判定情報に基づいて、各命令が前に実行される命令の完了を待つ必要があるか否かを判断し、この判断情報を各命令中に設定した命令列を作成する命令列生成手段と、

前記命令列生成手段により生成された命令列から命令を実行する前記計算機における並列実行可能な命令数単位で命令を取り出し、各命令それぞれに設定された上記判断情報に基づいて、各命令を順番に又は並列に実行させる命令実行手段とを備えたことを特徴とする命令実行方式。

【請求項２】　前記命令列生成手段により生成された命令列から命令を実行する前記計算機における並列実行可能な命令数単位で命令を取り出し、この取り出した各命令と前に実行される命令とを並列に実行するそれぞれの演算器を備えているか否かを検出するハザード検出手段を備え、前記命令実行手段は前記ハザード検出手段の検出情報に基づいて各命令を順番に又は並列に実行することを特徴とする請求項1記載の命令実行方式。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、主にＶＬＩＷ（Very Long Instruction Word：超長語命令）に代表されるような、命令を複数同時に並列実行する命令実行方式に関するものである。

【０００２】

【従来の技術】従来のＶＬＩＷにおいて、複数の命令を並列実行する命令数の決定方法は、単純に命令数を固定した固定長方式と命令数を指定しながら処理して行く可変長方式に大別できる。

【０００３】固定長方式は、並列実行が可能な限り、固定された命令数で構成されるＶＬＩＷの命令語を埋めて行くが、各命令によって汎用レジスタや演算器等が同時に使用されるリソース競合（例えばレジスタ競合、演算器競合等）によって並列実行不可能な命令の組み合わせが発生すると、リソース競合を避けるためにNOP命令（無効命令）をＶＬＩＷの命令語に挿入しなければならず、同時に実行される命令数がNOP命令の分だけ制限されることになる。この場合、NOP命令が多くなるほど、複数のＶＬＩＷの命令語からなる命令列が大きくなってしまい、その結果メモリを浪費しメモリの使用効率の悪い計算機システムになってしまう傾向がある。

【０００４】これに対処すべく、可変長方式や、固定長方式の場合にもＶＬＩＷの命令語中に命令の実行を指示

2

するための特別なフィールドを設けることによりメモリの使用効率の向上を図る考案がなされてきた。

【０００５】例えば、図３は、特開平9-26878に示された従来の固定長方式であり、並列実行可能な最大の命令数（以後、並列度と記載する）が３、即ち３つの命令でＶＬＩＷの１語が構成される場合の命令語フォーマットの例である。図３において、100a～100cはそれぞれ１つの命令を指定する命令フィールド、200は命令フィールド100a～100cで指定された命令の実行順番を指示する実行順指示フィールドである。この命令語フォーマットによれば、実行順指示フィールド200の内容に従って命令フィールド100a～100cで指定される３つの命令を同時に実行させたり任意の順番で実行させることが可能になり、NOP命令を使用する頻度を低減することができ、メモリの使用効率も向上する。

【０００６】しかし、この命令語フォーマットによる固定長方式では、並列度が増加するに従って実行順指示フィールド200のサイズが大きくなってしまい、実行順指示フィールド200の内容を解釈するためにH／Wを増加しなければならない。また、実行順指示フィールド200の内容を設定して命令列を生成するコンパイラ又はアセンブラでの処理も複雑になり、さらにNOP命令を使用しなければならない状況も依然残されている。

【０００７】また、この命令語フォーマットによる固定長方式では、各命令が、例えば加算、減算、乗算、積和等の同じ命令セットを使用していても、並列度が異なる複数の命令処理装置間では利用される命令語フォーマットが異なるため、命令列を再コンパイル又は再アセンブルしない限り、複数の命令処理装置間で互換性のある命令列を得ることができない。即ち命令列を再コンパイル又は再アセンブルしない限り命令列に互換性を得ることができない。

【０００８】さらに、例えば、図４は、特開平5-289870に示された従来の可変長方式の命令語フォーマットの例であり、図４において、300aは命令フィールド、400は命令語の終わりか否かを示す命令語終了フラグである。この命令語フォーマットによれば、例えば、命令処理装置の並列度が４であって、ある時点で演算器の制限から２つの命令のみしか実行させられない場合には、命令語終了フラグ400に終わりではないことを示す値を設定した命令と命令語終了フラグ400に終わりを示す値を設定した命令とをそれぞれ２つ連続で並べるだけでよく、NOP命令を排除することが可能である。

【０００９】しかし、この可変長方式では、並列度が小さいものから大きなものに変更された場合や、並列実行不可能な条件が緩やかな方向に変更された場合には、そのまま命令列を移植することが不可能ではないが、命令列を再コンパイル又は再アセンブルしない限り本来の命令処理性能を得ることができない。

【００１０】また、並列度が大きいものから小さいもの

への変更は、命令列が並列度に合わせて区切られるので、そのまま命令列を移植することが可能であるが、並列実行不可能な条件が厳しくなる方向に変更された場合には、全く動作不可能であり、命令列を再コンパイル又は再アセンブルしない限り命令列に互換性を得ることができない。

【００１１】さらに、分岐命令では必ず命令語が区切れると仮定するなど、命令語終了フラグ400のみでなく命令フィールド300aの内容も解釈しなければならず、その分処理が複雑である。

【００１２】上記以外の可変長方式としては、例えば、特開平8-161169に示されたＶＬＩＷの命令語中に、その命令語を構成する複数の命令フィールドのそれぞれが有効か否かを個別に示すビットを用意し、有効で無い場合には不要な命令フィールドを取り除いてメモリに保持しておき、実行時に元の命令語フォーマットに戻して実行するものがある。

【００１３】しかし、この可変長方式では複数の命令フィールドのそれぞれが有効か否かを判断するＨ／Ｗが必要になるため、Ｈ／Ｗを増加しなければならず、さらに並列度が異なる命令処理装置間では命令語フォーマットが異なるため、命令列を再コンパイル又は再アセンブルしない限り、複数の命令処理装置間で互換性のある命令列を得ることができない。即ち命令列を再コンパイル又は再アセンブルしない限り命令列に互換性を得ることができない。

【００１４】また、例えば、特開平5-233283に示された並列実行可能な命令数を指示する命令を命令語中に新規に挿入したものがあるが、並列実行できる命令数が変わる毎に前記指示命令を命令語の中に挿入しなければならないため、メモリの使用効率はあまり向上せず、Ｈ／Ｗの増加を招き、並列度が異なる場合には、再コンパイル等をしない限り命令列に互換性が得られない。

【００１５】一方、並列度が異なる命令処理装置間で命令列の互換性を得るという観点では、例えば、特開平8-241200又は特開平9-62507のような方式が考案されている。前者の場合には、並列度の高い命令列を並列度の低い命令処理装置で実行する場合には単純に命令列を分割して実行し、並列度の低い命令列を並列度の高い命令処理装置間で実行する場合には動作させる演算器を減らして命令列を実行することにしているが、再コンパイル又は再アセンブルしない限り本来の命令処理性能を得ることができない。

【００１６】また、後者の場合には、並列度の低い命令列を並列度の高い命令処理装置間で実行する場合に、複数のＶＬＩＷの命令語を同時に実行可能か否かをＨ／Ｗに判定させ、実行可能な場合に並列度を向上させようとするものであるが、単にスーパースカラ方式をＶＬＩＷの命令セットに適応させようとするものであり、多くのＨ／Ｗが必要となる。さらにこの方式では、各命令処理装置の並列度が整数倍の関係に無い限り動作させられない演算器が存在するため、再コンパイル又は再アセンブルしない限り本来の命令処理性能を得ることができない。

【００１７】

【発明が解決しようとする課題】以上のように、従来の命令の並列実行方式は、メモリの使用効率を向上させるために様々な命令語のフォーマットが提案されてきたが、各命令処理装置で実行できる命令列の互換性に乏しく、命令列の互換性を得るためには、再コンパイラが必要であり、またＨ／Ｗコストの増大や性能向上が必須となり、計算機システムとしてのスケーラビリティや機種展開をしていく上で大きな障害があった。この発明は、上記のような課題を解決するためになされたもので、Ｈ／Ｗコストを増大することなく、命令列を再コンパイルすることなく簡便な方法で命令列を並列実行することができ、またメモリの使用効率が良く、命令列の互換性を保証することができる命令実行方式を提供することを目的とする。

【００１８】

【課題を解決するための手段】第１の発明は、命令の並列実行が可能な複数の計算機の中で最大の並列実行可能命令数を入力する並列度指示手段と、前記並列度指示手段により入力された命令数と同数の命令単位で、各命令と各命令に先行する命令とが使用するレジスタの競合状態を判定する競合判定手段と、前記競合判定手段で判定されたレジスタ競合判定情報に基づいて、各命令が前に実行される命令の完了を待つ必要があるか否かを判断し、この判断情報を各命令中に設定した命令列を作成する命令列生成手段と、前記命令列生成手段により生成された命令列から命令を実行する前記計算機における並列実行可能な命令数単位で命令を取り出し、各命令それぞれに設定された上記判断情報に基づいて、各命令を順番に又は並列に実行させる命令実行手段とを備えたものである。

【００１９】第２の発明は、前記命令列生成手段により生成された命令列から命令を実行する前記計算機における並列実行可能な命令数単位で命令を取り出し、この取り出した各命令と前に実行される命令とを並列に実行するそれぞれの演算器を備えているか否かを検出するハザード検出手段を備え、前記命令実行手段は前記ハザード検出手段の検出情報に基づいて各命令を順番に又は並列に実行するものである。

【００２０】

【発明の実施の形態】実施の形態1. 一般に単純なコンパイラにおいては、まずプログラムを解析してシーケンシャルな命令列を用意し、その後並列度と、レジスタ、演算器などのＨ／Ｗリソースとの競合状況を判断しながら、前記シーケンシャルな命令列を先頭から順番に並列実行可能な命令フォーマットに変換して行く。このと

き、H／Wリソースの競合の代表的なものとして考えられるのは、汎用レジスタ等のレジスタや演算器が同時に使用されるレジスタ競合、演算器競合がある。

【００２１】ここで、演算器競合の場合、例えば、先行する演算命令の結果を後続の条件分岐命令が参照するようなものもこれに含むとして考えてもよいが、並列度には大きく左右されるものではなく、各命令が加算、減算、乗算等の同じ命令セットを使用して互換性が求められる計算機システム間においては、演算器が競合を起こす条件はほぼ同様と考えられる。

【００２２】一方、レジスタ競合の場合には、並列実行する単位で競合を判断する必要があるため、並列度によって得られる命令列が異なり、例えば、並列度２で前後２命令間でしかレジスタ競合を判定していない命令列をそのまま並列度４で実行すると、並列度４の中にはレジスタを同時に使用する命令が含まれる可能性もあり、レジスタの競合が起こりレジスタ衝突が発生して、正しくプログラムが動作しないことになる。このように、命令列に互換性を持たせるためにはレジスタ競合の方が演算器競合より影響力が大きい。

【００２３】そこで、本実施の形態１では、レジスタ競合に対しての解決方法について説明する。なお、実際にはレジスタ競合を判定するのみで、並列性が決定されるわけではないが、ここでは、説明の簡略化のためにその他の競合条件はないものとして説明を行う。

【００２４】図１は、この発明の命令実行方式の一実施例を示した図であり、この時の命令処理装置（計算機）の並列実行可能な命令数即ち並列度は３である。

【００２５】図１おいて、１ａ、１ｂ、１ｃ、…は前に実行される命令の完了を待つ必要があるか否かを示す完了待ちフィールド、２ａ、２ｂ、２ｃ、…はそれぞれ命令を指定する命令フィールド、３ａ、３ｂ、３ｃ、…は完了待ちフィールド！ａ、１ｂ、１ｃ、…および命令フィールド２ａ、２ｂ、２ｃ、…により構成される最小の命令実行単位である命令、４は複数の命令から構成される命令列、５は複数の命令処理装置の中で並列度の一番大きい値を入力する並列度指示手段、６は並列度指示手段５により入力された値に基づいて複数の命令のレジスタ競合の判定を行う競合判定手段、７は複数の命令処理装置（図示せず）上で実行させようとするプログラムを入力して命令列４を作成するコンパイラ、８は命令列４から複数の命令（この例では並列度が３であることから３つの命令）を取り出し、複数の命令のそれぞれに対応した完了待ちフィールドの内容に基づいて、複数の演算器（図示せず）に対して指示を与え、命令を並列実行させる命令実行手段である。

【００２６】次に、コンパイラ７において命令列４を生成するまでの動作について説明する。図１において、命令列の互換性を保つ必要のある複数の命令処理装置の中で並列度の一番大きなもの（最大の並列実行可能命令数）が例えば「４」であった場合、並列度指示手段５が「４」を入力する。並列度指示手段５が入力した「４」はコンパイラ７の持つ競合判定手段６に渡され、競合判定手段６ではこの「４」に基づいて、連続実行される可能性がある４つの命令間でレジスタの競合関係が判断されて、その判断結果が各命令の完了待ちフィールドに反映されて命令列４が生成される。

【００２７】つまりコンパイラ７では、ある１つの命令の完了待ちフィールドを設定するときに、その命令で使用するレジスタとその命令に先行する３命令で使用しているレジスタとが競合していないかを判断し、競合していればその完了待ちフィールドを「前に実行される命令の完了を待つ必要がある。」と設定し、競合がなければ「前に実行される命令の完了を待つ必要がない。」と設定する。このように各命令と先行する３命令とのレジスタ競合について判断することによって、並列実行していく全ての命令に対してこの完了待ちフィールドが設定される。

【００２８】この場合には、並列度指示手段５が入力した値より並列度が低い命令処理装置に対しては、レジスタ競合が判定される範囲が広いために、並列性が低く効率の悪い命令列となってしまう可能性があるが、元々この命令処理装置の並列度は低くて別の実行ステップになっている可能性が高いために問題はならないし、厳しい方向の条件でレジスタ競合が判定されているため、互換性が保証されることになる。

【００２９】一方、並列度指示手段５が入力した値と同じ値の並列度の命令処理装置に対しても、単純に全命令毎に先行する命令とのレジスタ競合を判定してしまうと、本来なら別の実行ステップとなるために競合と見做さなくてもよいものまで競合とされてしまう可能性があるが、一般の並列コンパイラで行われるような命令の並び換え等の技術を併用させれば特に問題とはならない。

【００３０】次に、命令実行手段８において命令を並列実行させるまでの動作について説明する。例えば、図１において、命令列４のうち命令Ｆ、Ｇ、Ｉのそれぞれの完了待ちフィールド１ｆ、１ｇ、１ｉのみに「前に実行される命令の完了を待つ必要がある。」と設定されていたとすると、命令実行手段８では、まず命令Ａ、Ｂ、Ｃを命令Ｘ、Ｙ、Ｚとして取り込み、命令Ｙ、Ｚの位置に来る命令Ｂ、Ｃの完了待ちフィールドが共に「前に実行される命令の完了を待つ必要がない。」ために、命令Ａ、Ｂ、Ｃを並列実行させる。

【００３１】次に、命令Ｄ、Ｅ、Ｆを命令Ｘ、Ｙ、Ｚとして取り込むが、命令Ｚの位置に来る命令Ｆの完了待ちフィールドが「前に実行される命令の完了を待つ必要がある。」ために、命令Ｄ、Ｅのみを並列実行させる。

【００３２】さらに、Ｆ、Ｃ、Ｈを命令Ｘ、Ｙ、Ｚとして取り込むが、命令Ｙの位置に来る命令Ｃの完了待ちフィールドが「前に実行される命令の完了を待つ必要があ

る。」ために、命令Ｆのみ実行させる。

【００３３】次に、命令Ｇ、Ｈ、Ｉを命令Ｘ、Ｙ、Ｚとして取り込むが、命令Ｚの位置に来る命令Ｉの完了待ちフィールドが「前に実行される命令の完了を待つ必要がある。」ために、命令Ｇ、Ｈのみ並列実行させる。

【００３４】なお、並列実行する命令処理装置の並列度が並列度指示手段５が入力した値と同じ「４」であれば、命令ＡＢＣＤ、命令Ｅ、命令Ｆ、命令ＧＨ、命令ＩＪＫＬ、というステップで実行が行われる。

【００３５】このように、並列実行する命令処理装置の並列度が「３」の例の場合には、命令ＡＢＣ、命令ＤＥ、命令Ｆ、命令ＧＨ、命令ＩＪＫ、命令Ｌ、…というように実行されて行くことになるが、コンパイラ７で生成された命令列４により並列性が保証されている部分（並列実行可能な命令部分）を分割して実行しているため、並列度指示手段５で指定する値を変更して再コンパイルしなくても命令列の互換性が保証されることになる。

【００３６】以上のように、本実施の形態によれば、命令毎に１ビットで済む完了待ちフィールド１ａ、１ｂ、１ｃ、…を用意し、完了待ちフィールド１ａ、１ｂ、１ｃ、…へは各命令処理装置中の最大の並列度でのコンパイル結果を反映させ、実行時に完了待ちフィールド１ａ、１ｂ、１ｃ、…が「前に実行される命令の完了を待つ必要がある。」ことを示していた場合には、先行する命令の実行が完了するのを確実に待ってから以降の命令を実行するようにしたので、並列度の異なる命令処理装置間での命令列の互換性を保つことができる。

【００３７】また、NOP命令が不要な可変長方式であるため、NOP命令によりメモリを浪費することもなくメモリの使用効率が高く、コンパイラ７および命令実行手段８での処理内容が特に複雑になることがないため、コスト増加を招くことのない計算機システムを得ることができる。

【００３８】なお、本実施の形態では、命令列９を生成する手段としてコンパイラを使用したものについて示したが、命令処理装置（図示せず）上で実行させようとするプログラムから命令列４を生成する命令列生成手段としては、アセンブラでも構わない。

【００３９】実施の形態２．実施の形態１では、並列度が異なるだけの命令処理装置しか存在しない場合を想定したものであるが、実際には、例えば、ある命令処理装置だけ他の命令処理装置とは異なり、加算命令と乗算命令は同時に実行できるものの、積和命令と加算命令、又は積和命令と乗算命令は同時に実行できないこともある。このような状況に対処するには、従来例と同様に再コンパイルするなどの方法も考えられるが、命令を並列実行させる時に並列実行が不可能な条件のみを判断してこの判断結果に基づいて処理を行うことができる。この一例について、本実施の形態２で説明する。

【００４０】図２は、この発明の命令実行方式の一実施例を示した図である。図において、９は命令Ｘ、Ｙ、Ｚに取り込まれた命令の命令フィールドの内容から並列実行不可能な条件を検出し、その検出結果を命令実行手段８に伝達するハザード検出手段である。ここで、コンパイラ７で命令列４を生成するまでの処理動作は実施の形態１と同様であり、ここでは説明を省略する。

【００４１】次に動作について説明する。例えば、図２において、命令Ｘ、Ｙ、Ｚに取り込まれた命令のうち命令Ｘが乗算命令、命令Ｚが積和命令であったとすると、元の命令列は乗算命令と積和命令が並列実行可能としてコンパイルされているために、レジスタ競合がなければ命令Ｚの位置に来る積和命令の持つ完了待ちフィールドは「前に実行される命令の完了を待つ必要がない。」と設定されており、そのままでは演算器の衝突が発生してしまい、正常に動作ができない。

【００４２】しかしこの場合には、ハザード検出手段９において、命令Ｘ、Ｙ、Ｚの命令フィールドから乗算命令と積和命令の組み合わせを検出する機能を用意し、その結果を命令実行手段８に反映させることで、命令ＸＹの実行と命令Ｚの実行のステップを分けることができ、命令列の互換性を保つことが可能になる。

【００４３】以上のように、本実施の形態によれば、他の命令処理装置と並列性が異なる部分の条件のみを検出する手段を設け、これに従って命令の実行を行うように処理することによって、並列性の異なる命令処理装置間での命令列の互換性を保つことができる。なお、ここで、他の命令処理装置と並列性が異なる命令処理装置を排除しておけば、本実施の形態に示すような処理は必要なく、実施の形態１のレジスタ競合を考慮した処理のみで並列度の異なる命令処理装置間での命令列の互換性を保つことができる。

【００４４】

【発明の効果】この発明は、以上発明したように構成されているので、以下に示すような効果を奏する。

【００４５】第１の発明では、命令の並列実行が可能な複数の計算機の中で最大の並列実行可能命令数と同数の命令単位で、各命令と各命令に先行する命令とが使用するレジスタの競合状態を判定し、この判定されたレジスタ競合判定情報に基づいて各命令が前に実行される命令の完了を待つ必要があるか否かを判断し、この判断情報を各命令中に設定し、命令実行時にこの各命令中の判断情報に基づいて各命令を順番に又は並列に実行するので、命令列中にNOP命令（無効命令）が不要になりメモリの使用効率が高く、また再コンパイルが不要であることからＨ／Ｗコストを増加することなく、並列度の異なる複数の計算機間で命令列の互換性を保った命令実行ができる。

【００４６】第２の発明では、並列実行可能な命令数単位で命令を取り出し、この取り出した各命令と前に実行

される命令とを並列に実行するそれぞれの演算器を備えているか否かの情報に基づいて各命令を順番に又は並列に実行するので、並列度の異なる複数の計算機間で命令列の互換性を保った命令実行ができる。

【図面の簡単な説明】

【図１】　実施の形態１の命令実行方式の説明図。

【図２】　実施の形態２の命令実行方式の説明図。

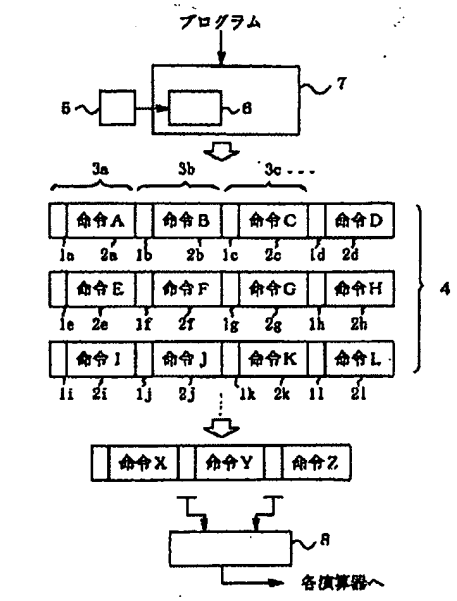
【図３】　従来の命令実行方式（固定長方式）の命令語フォーマットの構成図。

＊【図４】　従来の命令実行方式（可変長方式）の命令語フォーマットの構成図。

【符号の説明】

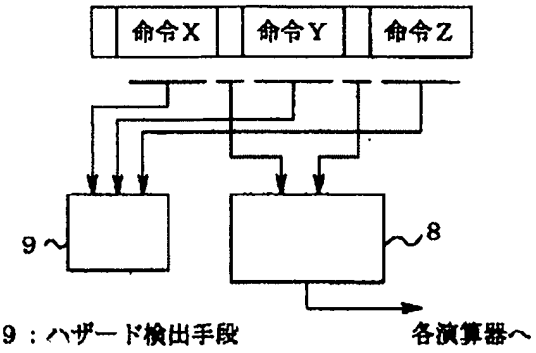
１ａ、１ｂ、１ｃ、…　完了待ちフィールド、２ａ、２ｂ、２ｃ、…　命令フィールド、３ａ、３ｂ、３ｃ、…　命令、４　命令列、５　並列度指示手段、６競合判定手段、７　コンパイラ、８　命令実行手段、９　ハザード検出手段。

【図１】

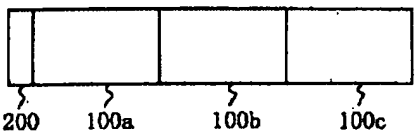


1a, 1b, 1c,… :完了待ちフィールド
2a, 2b, 2c,… :命令フィールド
3a, 3b, 3c,… :命令
4：命令列
5：並列度指示手段
6：競合判定手段
7：コンパイラ
8：命令実行手段

【図２】

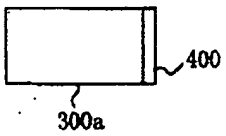


【図３】



100a, 100b, 100c:命令フィールド
200:実行順指示フィールド

【図４】



300a:命令フィールド
400:命令語終了フラグ